**<PROJECT 스펙보고서>**

**스마트 도어**

**팀원**

**2015104184 신은섭**

**2014104124이규태**

**교수님**

**김재홍**

Table of Contents

1. Abstract……………………………………………………………………………………… 3

2. Introduction……………………………………………………………………………….…3

3. BackgroundStudy…………………………………………………………………………..3

A. 관련 접근방법/기술 장단점 분석 3

B. 프로젝트 개발환경 3

4. Goal/Problem & Requirements……………………………………………………….3

5. Approach……………………………………………………………………………………...3

6. Project Architecture……………………………………………………………………….4

A. Architecture Diagram 4

B. Architecture Description 4

7. Implementation Spec……………………………………………………………………..4

A. Input/Output Interface 4

B. Inter Module Communication Interface 4

C. Modules 4

8. Solution………………………………………………………………………………………..4

A. Implementations Details 4

B. Implementations Issues 4

9. Results………………………………………………………………………………………….4

A. Experiments 4

B. Result Analysis and Discussion 4

10. Division & Assignment of Work……………………………………………………...4

11. Conclusion…………………………………………………………………………………….5

u [Appendix] User Manual………………………………………………………………...5

1. … 5

A. … 5

# **Abstract**

**문의 잠금을 풀때 비밀번호를 처야 해서 불편했던 순간이 누구나 한번 쯤은 있을 것이다. 손대지 않고 잠금을 풀기 위해 비콘과 스마트폰 어 플리케이션을 이용한 ‘스마트 도어’ 시스템을 소개한다.**

# **Introduction**

지금까지 도어락은 열쇠를 이용한 잠금에서부터 비밀번호를 이용하는 도어락으로 변하였다. 도어락이 변하면서 사람들은 더 편리하게 집을 잠글 수 있었다. 그러나 IOT가 활발히 사용되는 현재 상황에서도 아직까지 90년대에 나온 도어락은 사용하는 건 안좋다 생각한다. 그래서 우리는 자동으로 우리 집 인원만 판단하여 문을 열어주는 차세대 도어락을 제안 해야된다 생각했다. 우리가 제안하는 SMART DOOR는 어플과 IOT를 이용하는 도어락으로서, 가까이만 가면 잠금이 자동으로 해제되는 스마트 도어 시스템이다. 이를 통해 매일 들어가는 현관문, 번호키인 현관문을 열 때 번호조차 치기 귀찮을 때, 또, 두손 가득 장을 보고 돌아오면 현관문 비밀번호를 치기 위해 짐을 내려 놓아야 한때 등의 불편함을 해결할 수 있고 차세대 도어락을 제안할 수 있다.

# **Background Study**

## **관련 접근방법/기술 장단점 분석**

1. **스마트 폰을 이용한 사용자의 위치 탐색**

|  |  |  |  |
| --- | --- | --- | --- |
| **기술** | **설명** | **장점** | **단점** |
| **GPS** | **GPS위성의 정보를 받아 위치를 알려주는 시스템.** | **구현이 간단하다.** | **실내에서 잘 동작하지 않는다.**  **사용자가 GPS 기능을 켜놓아야 한다.** |
| **Bluetooth Beacon** | **Bluetooth Beacon에서 송신하는 신호를 받아 근처에 오면 정보를 알려주는 시스템.** | **실내, 근거리에서 GPS보다 오차가 작다.** | **먼 거리에서 오차가 크다. 사용자가 Bluetooth기능을 켜놓아야 한다.** |
| **고주파 근거리 탐지** | **고주파 소리를 내고 있으면 근처에오면 위치정보 서비스를 제공하는 시스템.** | **근거리에서 오차가 적다.**  **사용자가 다른 기능을 켜지 않아도 된다.** | **먼 거리에서 오차가 크다.**  **사용자가 마이크 접근을 허용해야 한다.** |

1. **라즈베리 파이와 스마트폰과의 무선통신**

**라즈베리 파이 3는 Bluetooth와 Wi-Fi를 내장하고 있기 때문에 두 가지 중에 한가지 방법을 선택하여 통신이 가능하다.**

## **프로젝트 개발환경**

|  |  |
| --- | --- |
| **IoT Board** | **Raspberry pi 3 b** |
| **IoT OS** | **Raspbian Stretch** |
| **Android OS** | **Android 8.0.0** |
| **Development Language** | **Python** |
| **IDE** | **VS Code** |
| **SECURITY** | **AWS Cognito** |

# **Goal/Problem & Requirements**

**등록된 휴대폰을 들고 문 근처로 가면 자동으로 잠금이 해제되는 문**

|  |  |
| --- | --- |
| **Requirements** | |
| **User Controller** | **Smart Phone** |
| **IOT Controller** | **Raspberry Pi** |
| **SeCurity & Server** | **AWS** |
| **IOT Device** | **Servor motor** |
| **Sensor Signal** | **Beacon** |

# **Approach**

**문에 부착된 장치가 블루투스 비콘 신호를 보내고 있다.**

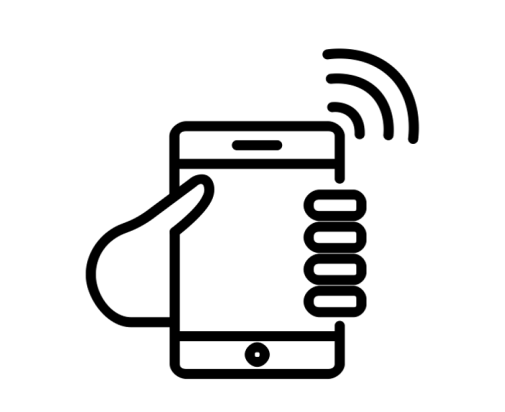
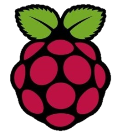
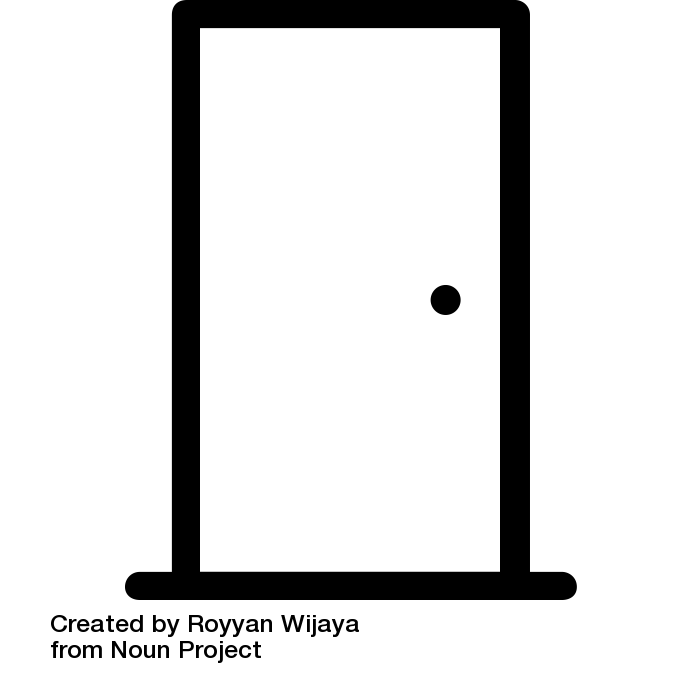
**어플리케이션이 주기적으로 신호를 감지 하고 있다.**

**신호 감지 중에 비콘 신호가 들어오면, AWS 서버에게 열림 신호를 전송한다.**

**신호를 받은 AWS가 적합한 사용자인지 인증 후에 해당 기기에게 열림 신호를 보내 문을 연다.**

# **Project Architecture**

## **Architecture Diagram**

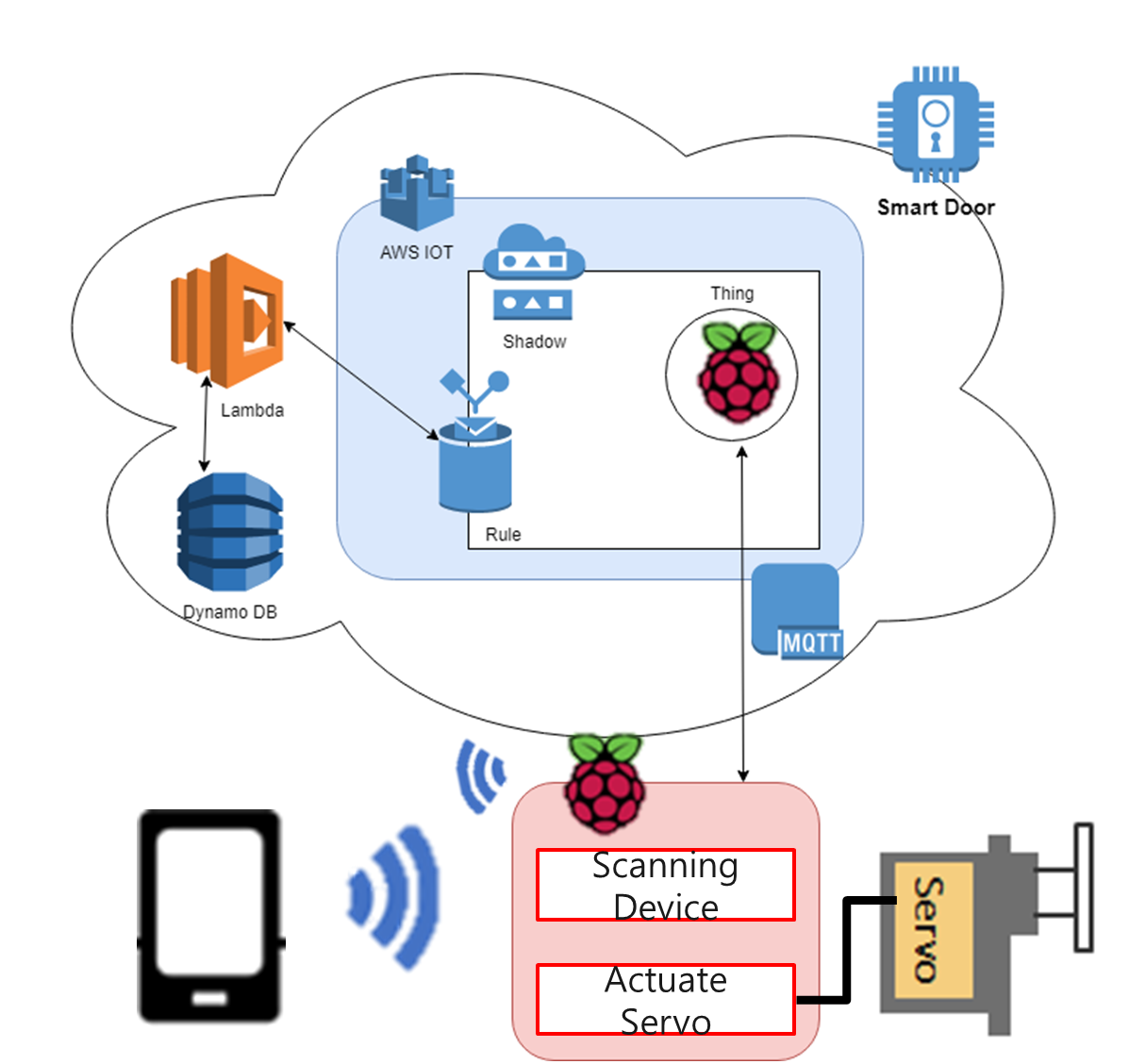


1

2

3

## **Architecture Description**

****

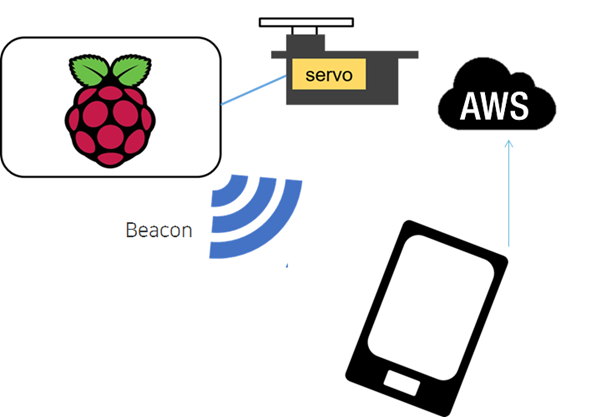
# **Implementation Spec**

1. Raspberry pi
2. **Bluetooth Beacon**
3. **Control Servo Motor**
4. **Recive ‘Open’ message and open door**
5. **Application**
6. **Scan Bluetooth beacon and Sent ‘Open’ message**
7. **AWS**

**1 Check User Signal and Device ID in DB.**

1. **Send ‘Open’ message to Raspberry Pi**

## **Input/Output Interface**



1. Application Input Signal : Beacon Signal

앱은 라즈베리 파이가 보내는 비콘신호를 수신 받는다. 비콘 신호에는 해당 라즈베리 파이의 맥주소와 RSSI(거리)가 전송되며, RSSI값이 특정 값 이내로 들어오면 앱이 실행된다.

1. Application Output Signal : ‘Open’ Message to AWS

라즈베리파이에 인접해오면 어플리케이션은 AWS에게 문을 열라는 ‘Open’ 신호를 보낸다. AWS에서는 해당 메시지를 받으면 메시지를 보낸 기기의 정보와 문을 열어야 하는 기기의 정보를 확인하고, 확인이 완료되면 해당 기기에게 문을 열어도 좋다는 메시지를 보낸다.

## **Inter Module Communication Interface**

애플리케이션에서 주기적으로 비콘 신호를 스캔하고, 등록된 기기의 비콘 신호가 들어왔을 경우에는 해당 비콘의 RSSI를 트래킹한다. RSSI값이 일정 범위안에 들어오면 문을 열라는 신호를 AWS에게 보낸다.

## **Modules**

Smart Door Device: 라즈베리파이가 mqtt를 이용해 smartdoor/[id]/open을 subscribe하고 있다가 smartdoor/[id]/open/accept으로 요청이 오면 moter를 작동시켜 문을 연다.

API Gateway : Smart Phone에서 문 제어와 관련된 모든 요청을 받고 lamda에게 넘겨서 처리하는 프록시이다.

Lambda : API Gateway의 프록시 람다이며, 실질적인 유저 조회 인증, 문 정보 조회, 상테체크 등의 작업을 수행한다.

# **Solution**

## **Implementations Details**

1. 사용자 인식

문에 달린 라즈베리파이가 지속적으로 비콘 신호를 송신합니다. 해당 비콘은 보안을 위해 별도의 내용을 포함하지는 않고 단순히 자신의 맥주소와 이름만을 송신합니다.

해당 라즈베리 파이가 등록된 스마트폰의 어플리케이션은 비콘신호를 받으면 RSSI 값을 이용해 비콘과의 거리를 추적합니다. 비콘과 인접하게 되면 RSSI값이 크게 증가하며, 이때 사용자는 AWS에게 문을 열라는 메시지를 보냅니다. (해당 어플리케이션을 사용하려면 AWS Cognito를 이용해서 사용자 인증을 거쳐야 앱을 사용할 수 있음)

1. 사용자 확인 및 문 열기

‘Open’메시지가 오면 AWS에서는 요청을 보낸 사용자 정보와 열어야 하는 기기정보를 확인합니다. 확인이 완료되면 해당 기기에게 문을 열라는 메시지를 보냅니다. 해당 메시지를 받은 기기는 문을 열기 위해 서보모터를 작동시켜 문 열림 버튼을 누릅니다.

1. **Door Thing ID Update**

**Smart Door Thing은 보안을 위해 일정 시간마다 ID를 업데이트 합니다. ID는 aws IoT Shadow를 통해 관리됩니다.**

|  |
| --- |
| **{**  **"reported": {**  **"state": "Active",**  **"doorid": "P789G8GP"**  **}**  **}** |

**기본적인 동작 상태의 Shadow는 위와 같습니다. Door Thing이 일정 주기로 ID를 요청합니다. 다음과 같은 상태로 바뀝니다.**

|  |
| --- |
| **{**  **"desired": {**  **"state": "RequireNewID",**  **},**  **"reported": {**  **"state": "RequireNewID",**  **"doorid": "P789G8GP"**  **}**  **}** |

**Desired의 State가 “RequestNewID”로 바뀌면 AWS IoT Role Engine이 다음과 같은 SQL로 상태변화를 감지합니다.**

|  |
| --- |
| SELECT \*, topic(3) as thingName  FROM '$aws/things/+/shadow/update'  WHERE state.desired.state = 'RequestNewID' |

**Role Engine이 상태변화를 감지하면 Lambda의 GenerateNewID 함수를 실행하여 IoT Shadow를 다음과 같이 업데이트 합니다.**

|  |
| --- |
| **{**  **"desired": {**  **"state": "Active",**  **"doorid": "CRWI5SCT"**  **},**  **"reported": {**  **"state": "Active",**  **"doorid": "P789G8GP"**  **}**  **}** |

**최종적으로 Door Thing이 이를 Shadow에 반영하여 ID가 업데이트 됩니다.**

|  |
| --- |
| **{**  **"reported": {**  **"state": "Active",**  **"doorid": "CRWI5SCT"**  **}**  **}** |

**모든 변경사항 로그는 AWS Cloud Watch에 기록됩니다.**

1. **Bluetooth와 Smart Phone 연결**

**라즈베리 파이가 주기적으로 신호를 보내 페어링 할 디바이스를 검색한다. 라즈베리 파이에 저장된 MAC과 스마트폰의 MAC이 동일할 경우 IOT를 작동시켜 문을 연다.**

**2-1 지속적인 연결을 시도하는 코드(5초 간격)**

|  |
| --- |
| **while True :**  **if target\_address != "None" :**  **target\_address = "None"**  **user = connect()**  **sleep(3)** |

**2-2 연결 대상을 페어링 하고, 페어링 완료 후 해당 기기의 정보와 라즈베리 파이에 저장된 정보를 확인한다.**

|  |
| --- |
| **def connect() :**  **nearby\_devices = discover\_devices()**  **#scanning for target device**  **for device\_id in nearby\_devices:**  **print(lookup\_name(device\_id))**  **if target\_name == lookup\_name(device\_id):**  **target\_address = device\_id**  **break**  **if target\_address is not None :**  **try:**  **sock.connect((target\_address, port))**  **print(target\_address)**  **#if check\_user(target\_address) == True :**  **# check\_signal(sock)**  **sock.close()**  **return True**  **except btcommon.BluetoothError as err:**  **print("An error occurred : %s " % err)**  **return False** |

## **Implementations Issues**

1. 신호 선별 Issue (Bluetooth vs Beacon)

처음에는 페어링된 사용자만 사용이 가능하도록 블루투스 신호를 이용하려 하였습니다. 그러나 블루투스 장치를 사용하기 위해서는 핸드폰 또는 라즈베리 파이가 주기적으로 상대방을 스캔해야 하는데 블루투스 장치 검색은 소모 전력량이 크다는 issue가 있었기 때문에 저전력을 사용하는 Beacon으로 바꾸었습니다. 그러나 Beacon의 경우는 저전력 이긴하지만 주위의 모든 기기에게 자신의 정보를 부리는 방식이기 때문에 단순히 자신의 위치정보와 핸드폰과의 거리를 측정하는 역할로만 사용하기로 하고 특별히 따른 보안신호를 고려하지는 않았습니다.

2. 어플리케이션 Service Issue

해당앱은 지속적으로 비콘을 스캔해야하는데 단순하게 앱을 설계하면 앱을 끄는 순간 스캔을 종료시켜버리는 critical issue가 발생하였습니다. 해당 문제를 해결하기 위해 앱을 나가도 꺼지지 않고, OS가 메모리 문제로도 해당 앱을 종료시키는 것을 방지하기 위해서 foregound Service로 앱을 제작하였습니다.

# **Results**

## **Experiments**

* 1. 데모 영상 확인

## **Result Analysis and Discussion**

* 1. 인증된 사용자가 인접거리 접근시 문이 열리는 프로젝트에서 가장 중요한 부분은 해결하였다. 그러나 애플리케이션 개발에 능숙하지 못해서 앱 UI를 잘 못꾸몄다는 점, 내부와 외부를 구별하여 내부에서는 문이 열리지 않도록 설정하기 위해 30분에 한번 GPS 정보를 비교하는 방법을 고안했지만 구현하지 못한 점, Cognito로 인증된 사용자만 들어갈 수 있도록 AWS 보안 시스템을 이용하였지만 집 문을 동작하는 만큼 추가적인 보안 강화 정책이 필요하다는 것이 이번 프로젝트에서 논의할 만한 주제인 것 같다.

# **Division & Assignment of Work**

|  |  |
| --- | --- |
| 항목 | 담당자 |
| Bluetooth Beacon | 이규태 |
| Control Servo Moter | 신은섭 |
| Recive ‘Open’ message and open door | 신은섭 |
| Scan Bluetooth beacon and Sent Open message | 이규태 |